Electronic Version
Stylesheet Version v1.1.1

Description

METHOD FOR USING SERIAL FLASH MEMORY AS PROGRAM STORAGE MEDIA FOR MICROPROCESSOR

BACKGROUND OF INVENTION

[0001] 1. Field of the Invention

[0002] The invention relates to a method for using a serial flash memory as the program storage media for a microprocessor, and more particularly, to apply only a serial flash or a serial flash together with a random access memory as the program storage media for a microprocessor.

[0003] 2. Description of the Prior Art

[0004] For microprocessor execution, it is necessary to provide instant programming code access. So that the data transmission speed between the microprocessor and the code storage memories is important. In the prior-art technique, we use parallel flash for code storage. When a microprocessor asks a program memory to provide data

(programming codes), an access location and other control signals corresponding to the desired programming code should be emitted from the microprocessor. Afterwards, the program memory, such as a read-only memory (ROM) or a flash memory, has to send the desired programming code(s) to the microprocessor within a certain period of time after the program memory receives the access location and the control signals from the microprocessor. Please refer to Fig.1, which is a schematic diagram showing how a microprocessor 10 accesses a program memory 12 according to the prior art. The microprocessor 10 emits an access address and a control signal towards the program memory 12, and the program memory 12 returns corresponding digital data (programming codes) back to the microprocessor 10 after receiving the access address and the control signals.

[0005] Regarding the evaluation of the performance of a memory, the most important part is the concern of access time and access speed. All of the processes, including the microprocessor emitting the access address and related control signals, the memory acquiring the access address data, the memory returning the corresponding digital data to the microprocessor, and the microprocessor actually re-

ceiving the desired digital data and finishing the data analyses, take an access cycle of the memory; that is, if the access cycle of the memory is set as 60ns, the whole desired time for executing the whole processes is 60ns. Please continue to refer to the prior-art embodiment shown in Fig.1. A complete access operation between the microprocessor 10 and the program memory 12 usually includes four following operating steps: (a) emitting the access address by the microprocessor 10, (b) waiting for the programming codes returned from the program memory 12, (c) analyzing the programming codes, and (d) emitting a new access address by the microprocessor 10. If the program memory 12 cannot finish the above-mentioned four operating steps within one access cycle, data loss, data clog, and inaccuracy of execution of the programming codes may occur.

[0006] The speed of the program memory 12 has to satisfy a certain requirement so that the microprocessor 10 can smoothly access and execute the programming codes in the program memory 12. Therefore, some memories, such as the serial flash memory, occupying less system sources and saving pin counts for the system at the expense of the access speed can not be applied in the prior-art struc-

ture installed with the microprocessor. Moreover, if we use a dynamic random access memory (DRAM) as the storage media of program memory, we must make sure the required instructions can feed back from the dynamic random access memory in certain microprocessor cycles (usually a memory access cycle), so that we may need very high speed DRAM for satisfying the requirement all the time.

[0007] In addition, a typical simulation process has to reflect the practical operational conditions of the original system. According to the simulation results, some possible errors may be rectified in advance. In general, an in-circuit emulator (ICE) duplicates and imitates the behavior of a chip and the in-circuit emulator emulates by using programming techniques and special machine features to permit executing codes written for the chip that it imitates. In brief, the in-circuit emulator is a hardware component used for emulating behaviors of the microprocessor circuit and externally connected to the original microprocessor system as an expansion of the original microprocessor system. With the in-circuit emulator, designers can perform debug-simulating operations for the microprocessor system. Please refer to Fig.2, which is a schematic dia-

gram showing an in-circuit emulator 24 emulating a microprocessor system 20 according to the prior art. As shown in Fig.2, a chip 20 (the microprocessor system), an in-circuit emulator 24, a program memory 22, and an external clock generator 26 are included. Please refer to both Fig.1 and Fig.2. The in-circuit emulator 24 replaces the chip 20 (the microprocessor) to emulate actual operations of the chip 20, which includes the microprocessor 10 shown in Fig.1.In the prior-art embodiment, an operating clock of the in-circuit emulator 24 is provided from the external clock generator 26 having no relation with the emulated chip 20 (the microprocessor system 20). The program memory 22 can be used to provide instructions for operating the in-circuit emulator 24, and the in-circuit emulator 24 can be used to provide the access address and related control signals toward the emulated chip. Afterwards, the chip returns the corresponding digital data to the in-circuit emulator 24 according to the access address and the related control signals. The in-circuit emulator duplicates and imitates the behavior of a chip according to the above-mentioned description.

[0008] However, by either utilizing the in-circuit emulator or executing some present simulation software, actual operat-

ing conditions of the microprocessor system 20 with a serial flash memory are still hard to imitate. Because when a cost-effective, simple, and slow serial flash memory, is integrated with a high-speed microprocessor, such as the high-speed microprocessor system 20 shown in Fig.2, the fluctuation of the clock, interruption, and suspension of operation should be taken into consideration of design. According to the prior art, on the premise that it is almost impossible to use a (external) clock generator 26 to imitate those dynamic operations, it is needless to consider the possibility of the integration between the low-speed memory and the high-speed microprocessor system.

SUMMARY OF INVENTION

[0009] It is therefore a primary objective of the claimed invention to provide a method for using a serial flash memory or a serial flash memory together with a random access memory as program memory to provide the instructions of a microprocessor. We can dynamically adjusting the operating speed of the microprocessor to access the serial flash memory (together with the random access memory) and to solve the above-mentioned problems.

[0010] In the embodiments according to the present invention, operating (executing) speed of a microprocessor is ar-

ranged by adjusting an operating clock. We set a buffering/controlling device in the microprocessor system for outputting the operating clock to the microprocessor and for consecutively accessing a predetermined number of data (programming codes) from a serial flash memory (together with a random access memory). When the microprocessor requires data, it will first check whether the buffering/controlling device stores the desired data (programming codes) of the microprocessor. If the buffering/controlling device stores the desired data, the microprocessor accesses the desired data directly from the buffering/controlling device; on the other hand, the buffering/controlling device will slow down or temporarily stop the operating clock, so that the microprocessor will suspend and retain the current conditions due to the disappearance of the operating clock. After the serial flash memory (together with the random access memory)searches and returns the desired programming codes of the microprocessor, the buffering/controlling device will recover the operating clock. Therefore, the operating clock can be dynamically controlled, and using the serial flash memory (together with the random access memory) as program memory is possible.

[0011] In the present invention, we further disclose a technique for dynamically adjusting an operating clock of a microprocessor emulator. The microprocessor emulator is electrically connected to a buffering/controlling device, and the buffering/controlling device outputs the operating clock to the microprocessor emulator for operating the microprocessor emulator. Therefore, the buffering/controlling device can dynamically adjust the operating clock according to whether the access address emitted from the microprocessor emulator is located in the buffering/controlling device. The mechanism of the present invention can emulate the behavior of a microprocessor to access a low-speed serial flash memory by utilizing the buffering/controlling device.

[0012] According to the claimed invention, we disclosure a method for dynamically adjusting an operating speed of a microprocessor for the microprocessor to access at least a serial flash memory (together with a random access memory) including reducing an executing speed of the microprocessor if data in the serial flash memory (together with the random access memory) is not well prepared and executing the microprocessor at a normal speed if data in the serial flash memory (together with the random access

memory) is well prepared.

[0013] According to the claimed invention, we disclosure a method for dynamically adjusting an operating speed of a microprocessor emulator, the emulator can emulate the behavior of microprocessor system with a serial flash memory (together with a random access memory).

[0014] These and other objectives of the present invention will no doubt become obvious to those of ordinary skill in the art after reading the following detailed description of the preferred embodiment, which is illustrated in the various figures and drawings.

Brief Description of Drawings

[0015] Fig.1 is a schematic diagram showing how a microprocessor accesses a program memory according to the prior art.

[0016] Fig.2 is a schematic diagram showing an in-circuit emulator emulating a microprocessor system according to the prior art.

[0017] Fig.3 is a functional block diagram of an embodiment according to the present invention.

[0018] Fig.4 is a schematic diagram of a detailed embodiment shown in Fig.3.

[0019] Fig.5 is a timing chart including a covering mask signal,

the operating clock, the access address, and the programming codes.

[0020] Fig.6 is a flowchart of an embodiment according to the present invention.

[0021] Fig.7 is a functional block diagram of another embodiment according to the present invention.

[0022] Fig.8 is a schematic diagram of a detailed embodiment according to the present invention.

[0023] Fig.9 is a block diagram of a processing system according to another embodiment of the present invention.

[0024] Fig.10 is a block diagram of a processing system according to another embodiment of the present invention.

DETAILED DESCRIPTION

[0025] Several preferred embodiments according to the present invention are introduced as follows. The method includes reducing an executing speed of the microprocessor if data in the serial flash memory (together with the dynamic random access memory) is not well prepared and executing the microprocessor at a normal speed if data in the serial flash memory (together with the dynamic random access memory) is well prepared. The buffering/controlling device can be replaced by a microprocessor-operating-speed control device. The following embodiments uti-

lizing the buffering/controlling device are only examples according to the present invention.

[0026] Please refer to Fig.3, which is a functional block diagram of an embodiment according to the present invention. The structure shown in Fig.3 includes a buffering/controlling device 38, a microprocessor 30, and a serial flash memory 32. The serial flash memory 32 stores a plurality of digital instructions. In contrast with the prior-art embodiment shown in Fig.1, during the access process, the serial flash memory 32 in the present embodiment may not support enough bit rate (programming codes access rate) that microprocessor 30 needed. When the microprocessor 30 operates at a high speed, we can use the present invention to help the microprocessor 30 access the (low-speed)serial flash memory 32 via the buffering/controlling device 38 for avoiding data loss or data clog.

[0027] Please continue to refer to Fig.4. In the present embodiment, the microprocessor 30 does not directly acquire the desired digital data from the serial flash memory 32 but utilizes the buffering/controlling device 38 beforehand to consecutively access a plurality of digital data in the serial flash memory 32 instead. The buffering/controlling device 38 outputs an operating clock to the microprocessor 30

so as to slow down/stop microprocessor 30; that is, when the operating clock disappears, the operations of the microprocessor 30 will suspend, when the operating clock slow down, the operations of the microprocessor 30 will also slow down. The buffering/controlling device 38 can consecutively access a predetermined number of programming codes in the serial flash memory 32. When the microprocessor 30 requires the programming codes and accesses the buffering/controlling device 38, the microprocessor 30 emits an access address corresponding to the programming codes to the buffering/controlling device 38 so that the buffering/controlling device 38 can judge whether the access address emitted from the microprocessor 30 is located in the buffering/controlling device 38. If the buffering/controlling device 38 stores the programming codes required by the microprocessor 30, the microprocessor 30 directly receives the desired programming codes from the buffering/controlling device 38. On the other hand, if the buffering/controlling device 38 does not store the programming codes required by the microprocessor 30 (for instance, the microprocessor 30 is in jump condition), the buffering/controlling device 38 will stop the outputting of the operating clock so that the

microprocessor 30 will suspend and retain the current conditions due to the disappearance of the operating clock. In the meantime, the buffering/controlling device 38 transmits the access address corresponding to the programming codes to the serial flash memory 32. After receiving the access address, the serial flash memory 32 will search and return the searched programming codes to the buffering/controlling device 38 and the microprocessor 30(for fasten the whole access process). The buffering/controlling device 38 then recovers the operating clock for the microprocessor 30 so that the microprocessor 30 can access the programming codes. With the structure and method according to the present embodiment, the microprocessor 30 can execute the codes when accessing the low-speed serial flash memory 32.

[0028] Please refer to Fig.10. In the present embodiment, the program is stored in the random access memory 34(it's usually a dynamic random access memory). We first load the instructions from the serial flash memory 32 to the random access memory 34,then the microprocessor 30 will execute the program via the random access memory 34. The microprocessor 30 does not directly acquire the desired digital data from the random access memory 34

but utilizes the buffering/controlling device 38 beforehand to consecutively access a plurality of digital data in the random access memory 34instead. The buffering/controlling device 38 outputs an operating clock to the microprocessor 30 so as to slow down/stop microprocessor 30; that is, when the operating clock disappears, the operations of the microprocessor 30 will suspend, when the operating clock slow down, the operations of the microprocessor 30 will also slow down. The buffering/controlling device 38 can consecutively access a predetermined number of programming codes in the random access memory 34. When the microprocessor 30 requires the programming codes and accesses the buffering/controlling device 38, the microprocessor 30 emits an access address corresponding to the programming codes to the buffering/controlling device 38 so that the buffering/controlling device 38 can judge whether the access address emitted from the microprocessor 30 is located in the buffering/controlling device 38. If the buffering/controlling device 38 stores the programming codes required by the microprocessor 30, the microprocessor 30 directly receives the desired programming codes from the buffering/controlling device 38. On the other hand, if the

buffering/controlling device 38 does not store the programming codes required by the microprocessor 30 (for instance, the microprocessor 30 is in jump condition), the buffering/controlling device 38 will stop the outputting of the operating clock so that the microprocessor 30 will suspend and retain the current conditions due to the disappearance of the operating clock. In the meantime, the buffering/controlling device 38 transmits the access address corresponding to the programming codes to the random access memory 34. After receiving the access address, the random access memory 34 will search and return the searched programming codes to the buffering/controlling device 38 and the microprocessor 30(for fasten the whole access process). The buffering/controlling device 38 then recovers the operating clock for the microprocessor 30 so that the microprocessor 30 can access the programming codes. With the structure and method according to the present embodiment, the microprocessor 30 can execute the codes when accessing the low-speed random access memory 34.

[0029] Please continue to refer to Fig.9. In the present embodiment, the program is partially stored in the random access memory 34(it's usually a dynamic random access

memory) and partially stored in the serial flash memory 32. We first load partial of the instructions from the serial flash memory 32 to the random access memory 34,then the microprocessor 30 will execute the program via the random access memory 34 or the serial flash memory 32.Usually, for faster response speed, we will put frequently used codes into random access memory 34, and if the required instructions are available in the random access memory 34, the buffering/controlling device 38 will fetch the codes from the random access memory 34, else the buffering/controlling device 38 fetch the codes from the serial flash memory 32.In fact, the microprocessor 30 does not directly acquire the desired digital data from the random access memory 34 or the serial flash memory 32 but utilizes the buffering/controlling device 38 beforehand to consecutively access a plurality of digital data in the random access memory 34 or the serial flash memory 32instead. The buffering/controlling device 38 outputs an operating clock to the microprocessor 30 so as to slow down/stop microprocessor 30; that is, when the operating clock disappears, the operations of the microprocessor 30 will suspend, when the operating clock slow down, the operations of the microprocessor 30 will also slow down.

The buffering/controlling device 38 can consecutively access a predetermined number of programming codes in the random access memory 34 or the serial flash memory 32. When the microprocessor 30 requires the programming codes and accesses the buffering/controlling device 38, the microprocessor 30 emits an access address corresponding to the programming codes to the buffering/controlling device 38 so that the buffering/controlling device 38 can judge whether the access address emitted from the microprocessor 30 is located in the buffering/controlling device 38. If the buffering/controlling device 38 stores the programming codes required by the microprocessor 30, the microprocessor 30 directly receives the desired programming codes from the buffering/controlling device 38. On the other hand, if the buffering/controlling device 38 does not store the programming codes required by the microprocessor 30 (for instance, the microprocessor 30 is in jump condition), the buffering/controlling device 38 will stop the outputting of the operating clock so that the microprocessor 30 will suspend and retain the current conditions due to the disappearance of the operating clock. In the meantime, the buffering/controlling device 38 transmits the access address

corresponding to the programming codes to the random access memory 34 or the serial flash memory 32. After receiving the access address, the random access memory 34 or the serial flash memory 32 will search and return the searched programming codes to the buffering/controlling device 38 and the microprocessor 30(for fasten the whole access process). The buffering/controlling device 38 then recovers the operating clock for the microprocessor 30 so that the microprocessor 30 can access the programming codes. With the structure and method according to the present embodiment, the microprocessor 30 can execute the codes when accessing the low-speed random access memory 34 or the serial flash memory 32.

[0030] In the embodiments of the present invention, the buffering/controlling device 38 can stop outputting the operating clock by marking the operating clock. The mechanism can be achieved by setting a covering mask signal in the buffering/controlling device 38. Please refer to Fig.5, which is a timing chart showing a covering mask signal, the operating clock, the access address, and the programming codes. Please refer to Fig.5. When the access address A2 (corresponding to the programming code C2) required by the microprocessor 30 is not located in the

buffering/controlling device 38, the covering mask signal will be raised to a predetermined voltage level to stop the operating clock. In the meanwhile, the microprocessor 30 retains its current condition (sending the access address A2) because missing of operating clock. When the desired programming code C2 of the microprocessor 30 returns from the serial flash memory 32 or the random access memory 34 to the buffering/controlling device 38 and the microprocessor 30, the covering mask signal will be restored to another predetermined voltage level to release the operating clock and to continue the operations of the microprocessor 30. A front (rising) edge of the covering mask signal represents that the access address (and the corresponding programming codes) of the microprocessor 30 is not located in the buffering/controlling device 38, while the rear (falling) edge of the covering mask signal represents the required programming codes of the microprocessor 30 is found in the serial flash memory 32 or the random access memory 34 and already captured by the buffering/controlling device 38 (and the microprocessor 30). Therefore, by utilizing the buffering/controlling device 38 combined with dynamically adjustable operating clock according to the present invention, the high-speed

microprocessor 30 can smoothly access the low-speed serial flash memory 32 or the random access memory 34.

[0031] In summary, the present invention utilizes a buffering/controlling device and a covering mask signal to dynamically adjust the operating clock of a microprocessor, so that the microprocessor can smoothly access a memory. Please refer to Fig.6, which is a flowchart of an embodiment according to the present invention.

[0032] Step 100: Begin;

[0033] Step 102: Utilize the buffering/controlling device to output the operating clock to the microprocessor so as to control the microprocessor;

[0034] Step 104: Utilize the buffering/controlling device to access a predetermined number of digital data stored in the serial flash memory or the random access memory . In the embodiment shown in Fig.3, Fig.4, Fig.9 and Fig.10, the buffering/controlling device can consecutively access the predetermined number of programming codes at a starting address in the serial flash memory or the random access memory;

[0035] Step 106: Utilize the microprocessor to access desired digital data from the buffering/controlling device and utilize the buffering/controlling device to judge whether the

desired digital data (corresponding to the access address) of the microprocessor are located in the buffering/controlling device. If the desired digital data are located in the buffering/controlling device, proceed with Step 112; If the microprocessor desired digital data are not located in the buffering/controlling device, proceed with Step 108.

[0036] Step 108: Utilize the buffering/controlling device to stop outputting the operating clock to suspend the microprocessor and to retain current conditions of the microprocessor. For instance, a covering mask signal will be raised to a predetermined voltage level to stop the operating clock. In the meantime, the buffering/controlling device transmits the access address and the control signals to the serial flash memory or the random access memory;

[0037] Step 110: After the serial flash memory or the random access memory receives related control signals and the access address corresponding to the required codes, the serial flash memory or the random access memory searches and returns the searched digital data (the programming codes) to the buffering/controlling device and the microprocessor. The buffering/controlling device release the operating clock for the microprocessor (lowering the covering mask signal to a predetermined voltage level), so

that the microprocessor can continue to operate;

[0038] Step 112: Continue to proceed with normal data access operations; that is, utilize the microprocessor to continue accessing the desired digital data (the programming codes) from the buffering/controlling device. Go back to Step 106 to process additional data.

[0039] Regarding the emulation of the microprocessor according to the present invention, the above-mentioned characteristics of method and structure are still suitable. Please refer to Fig.2 (the prior-art embodiment), Fig.3, and Fig.4. In order to access a low-speed serial flash memory in the structure of the present invention, an additional buffering/controlling device 38 is installed between the microprocessor 30 and the serial flash memory 32. When the programming code required by the microprocessor 30 is not located in the buffering/controlling device 38, the operating clock will be stopped and the microprocessor 30 will retain its current condition. When the desired programming code of the microprocessor 30 returns from the serial flash memory 32 to the buffering/controlling device 38 and the microprocessor 30, the operating clock will be released and the microprocessor 30 will continue to operate. Since in real circumstances the operating clock of the

microprocessor 30 does not regularly appear, the emulator according to the prior art shown in Fig.2 can imitate neither the dynamic situation nor the mechanism that the buffering/controlling device.

[0040] Please refer to Fig.7, which is a functional block diagram of another embodiment according to the present invention. The present embodiment includes a buffering/controlling device 58 and a microprocessor emulator 54. The buffering/controlling device 58 is installed in a microprocessor system 50, and the buffering/controlling device 58 and the microprocessor emulator 54 are mutually connected. Concerning the previous embodiments shown in Fig.3 and Fig.4, the microprocessor emulator 54 corresponds to the microprocessor 30. In order to make the microprocessor emulator 54 authentically emulate the operation of the microprocessor system 50 of the present invention, the buffering/controlling device 58 provides an operating clock to the microprocessor emulator 54 so as to control the microprocessor emulator 54. Whether the buffering/controlling device 58 originally stores a predetermined number of address data, the microprocessor emulator 54 emits an access address to the buffering/controlling device 58 when starting to perform emulating

operation. When the access address is located in the buffering/controlling device 58, the buffering/controlling device 58 continues to output the operating clock to the microprocessor emulator 54 to maintain operations of the microprocessor emulator 54. When the access address is not located in the buffering/controlling device 58, the buffering/controlling device 58 will stop outputting the operating clock to the microprocessor emulator 54 so as to suspend the microprocessor emulator 54. Therefore, utilizing the buffering/controlling device 58 to provide with the dynamically adjustable operating clock for the microprocessor emulator 54 directs an effective way to dynamically control the microprocessor emulator 54 and to accurately emulate the microprocessor system 50 with characteristics of the present invention.

[0041] Another approach for emulation of the present embodiment is that the buffering/controlling device 58 automatically recovers to output the operating clock to the microprocessor emulator 54 in order to recover the operations of the microprocessor emulator 54 after a predetermined number of the operating clock cycles pass (the operating clock starts to operate after the buffering/controlling device 58 stops to output the operating clock to suspend the

microprocessor emulator 54). Since the emulation process is still different from the actual operation, the connection between a memory and the buffering/controlling device 58 does not matter. If the buffering/controlling device 58 is electrically connected to a low-speed serial flash memory stored with a plurality of digital data, the whole structure (including the buffering/controlling device 58 and the low-speed memory) is almost equal to the embodiment of the present invention shown in Fig.3 and Fig.4. Please refer to Fig.8, which is a schematic diagram of a detailed embodiment according to the present invention. The buffering/controlling device 58 is implemented with a FIFO storage structure and electrically connected to a low-speed serial flash memory 52. The microprocessor emulator 54 is an in-circuit emulator 24. The buffering/controlling device 58 will consecutively access a predetermined number of digital data and corresponding access addresses at a starting address of the serial flash memory 52; that is, those digital data and corresponding access addresses are stored beforehand to the buffering/controlling device 58. When the access address emitted from the microprocessor emulator 54 is located in the buffering/controlling device 58, the buffering/controlling device

58 will deliver the digital data corresponding to the access address to the microprocessor emulator 54. When the access address is not located in the buffering/controlling device 58 (for example, the jump condition executed by the microprocessor 30), the serial flash memory 52 will search and return digital data corresponding to the access address back to the buffering/controlling device 58 and the buffering/controlling device 58 then delivers the searched digital data to the microprocessor emulator 54. In the meantime, the operating clock recovers. Certainly, if the microprocessor emulator 54 is electrically connected to the serial flash memory 52, serial flash memory 52 can directly return the digital data to the microprocessor emulator 54 without any intermediary device such as the buffering/controlling device 58 immediately after the serial flash memory 52 searches the desired digital data corresponding to the access address.

[0042] Please notice that, in the present embodiment, the buffering/controlling device 58 can stop outputting the operating clock by covering/marking the operating clock. Therefore, a covering mask signal should be included in the buffering/controlling device 58. When the desired access address of the microprocessor emulator is not located in

the buffering/controlling device 58, the buffering/controlling device 58 will raise the value of the covering mask signal to a predetermined voltage level so as to cover the operating clock. After a predetermined number of clock cycles (or after the digital data begin to be returned from the serial flash memory 52 to the buffering/controlling device 58 and the microprocessor emulator 54), the covering mask signal will be recovered to an initial predetermined low voltage level so as to recover the operating clock and to operate the microprocessor emulator 54.

[0043] Please continue to refer to Fig.8. The microprocessor emulator 54 is electrically connected to a second memory 53, which can be a program memory such as a static random access memory (SRAM) or a ROM. A plurality of instructions, which are required for operations of the microprocessor emulator 54, are stored in the second memory 53. When the buffering/controlling device 58 outputs the operating clock to the microprocessor emulator 54, the second memory 53 will deliver related instructions to the microprocessor emulator 54. When the buffering/controlling device 58 suspends to output the operating clock to the microprocessor emulator 54, the microprocessor emulator 54 suspends and is unable to receive any instruction from

the second memory 53. In addition, the frequency of the operating clock provided by the buffering/controlling device 58 can be adjusted according to the real situations. The frequency of the operating clock can also be adjusted by being electrically connected to an external clock generator 56. In summary, the microprocessor emulator 54 of the present invention can accurately emulate the performances of the microprocessor system 50 combined with the low-speed serial flash memory 52 and a dynamically adjustable operating clock.

[0044] Those skilled in the art will readily observe that numerous modifications and alterations of the device and method may be made while retaining the teachings of the invention. Accordingly, the above disclosure should be construed as limited only by the metes and bounds of the appended claims.